**Neural Networks & Deep Learning - Assignment-9**

**Question-1:**

**Video link-https://drive.google.com/file/d/128Ye3Bb2hLxPlv75GpY1e6gYAczVdbNl/view?usp=sharing**

import pandas as pd

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

from keras.models import Sequential

from keras.layers import Dense, Embedding, LSTM

from sklearn.model\_selection import train\_test\_split

from keras.utils import to\_categorical

import re

from sklearn.preprocessing import LabelEncoder

# Load the data

data = pd.read\_csv('/content/Sentiment (3) (1).csv')

data = data[['text','sentiment']]

# Preprocess the text data

data['text'] = data['text'].apply(lambda x: x.lower())

data['text'] = data['text'].apply(lambda x: re.sub('[^a-zA-z0-9\s]', '', x))

for idx, row in data.iterrows():

    row[0] = row[0].replace('rt', ' ')

# Tokenize and pad sequences

max\_features = 2000

tokenizer = Tokenizer(num\_words=max\_features, split=' ')

tokenizer.fit\_on\_texts(data['text'].values)

X = tokenizer.texts\_to\_sequences(data['text'].values)

X = pad\_sequences(X)

# Define the model architecture

embed\_dim = 128

lstm\_out = 196

def create\_model():

    model = Sequential()

    model.add(Embedding(max\_features, embed\_dim, input\_length=X.shape[1]))

    model.add(LSTM(lstm\_out, dropout=0.2, recurrent\_dropout=0.2))

    model.add(Dense(3, activation='softmax'))

    model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model

# Encode the target labels

label\_encoder = LabelEncoder()

integer\_encoded = label\_encoder.fit\_transform(data['sentiment'])

y = to\_categorical(integer\_encoded)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

# Train the model

batch\_size = 32

model = create\_model()

model.fit(X\_train, y\_train, epochs=1, batch\_size=batch\_size, verbose=2)

# Evaluate the model

score, acc = model.evaluate(X\_test, y\_test, verbose=2, batch\_size=batch\_size)

print("Test score:", score)

print("Test accuracy:", acc)

# Save the model

model.save('sentiment\_analysis\_model.h5')

# Load the saved model

from keras.models import load\_model

loaded\_model = load\_model('sentiment\_analysis\_model.h5')

# New text data for prediction

new\_text = "A lot of good things are happening. We are respected again throughout the world, and that's a great thing."

new\_text = new\_text.lower()

new\_text = re.sub('[^a-zA-z0-9\s]', '', new\_text)

new\_text\_sequence = tokenizer.texts\_to\_sequences([new\_text])

new\_text\_sequence = pad\_sequences(new\_text\_sequence, maxlen=X.shape[1])

# Make predictions on new data

prediction = loaded\_model.predict(new\_text\_sequence)

sentiment\_labels = ['negative', 'neutral', 'positive']

predicted\_sentiment = sentiment\_labels[prediction.argmax()]

print("Predicted sentiment:", predicted\_sentiment)

**output:**

291/291 - 43s - loss: 0.8254 - accuracy: 0.6425 - 43s/epoch - 149ms/step

144/144 - 4s - loss: 0.7606 - accuracy: 0.6772 - 4s/epoch - 27ms/step

Test score: 0.7606064677238464

Test accuracy: 0.677151620388031

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')`.

saving\_api.save\_model(

1/1 [==============================] - 0s 262ms/step

Predicted sentiment: negative

**Question-2:**

**Video link -https://drive.google.com/file/d/1nG0xRoFVeou3u0caLisuFXS-cd\_T1CaW/view?usp=sharing**

import pandas as pd

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

from keras.models import Sequential

from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D

from matplotlib import pyplot

from sklearn.model\_selection import train\_test\_split

from keras.utils.np\_utils import to\_categorical

import re

from sklearn.preprocessing import LabelEncoder

from keras.wrappers.scikit\_learn import KerasClassifier

from sklearn.model\_selection import GridSearchCV

data = pd.read\_csv('data/Sentiment.csv')

data = data[['text','sentiment']]

data['text'] = data['text'].apply(lambda x: x.lower())

data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))

for idx, row in data.iterrows():

    row[0] = row[0].replace('rt', ' ')

max\_fatures = 2000

tokenizer = Tokenizer(num\_words=max\_fatures, split=' ')

tokenizer.fit\_on\_texts(data['text'].values)

X = tokenizer.texts\_to\_sequences(data['text'].values)

X = pad\_sequences(X)

embed\_dim = 128

lstm\_out = 196

def createmodel():

    model = Sequential()

    model.add(Embedding(max\_fatures, embed\_dim,input\_length = X.shape[1]))

    model.add(LSTM(lstm\_out, dropout=0.2, recurrent\_dropout=0.2))

    model.add(Dense(3,activation='softmax'))

    model.compile(loss = 'categorical\_crossentropy', optimizer='adam',metrics = ['accuracy'])

    return model

labelencoder = LabelEncoder()

integer\_encoded = labelencoder.fit\_transform(data['sentiment'])

y = to\_categorical(integer\_encoded)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,y, test\_size = 0.33, random\_state = 42)

batch\_size = 32

model = createmodel()

model.fit(X\_train, Y\_train, epochs = 1, batch\_size=batch\_size, verbose = 2)

score,acc = model.evaluate(X\_test,Y\_test,verbose=2,batch\_size=batch\_size)

print(score)

print(acc)

print(model.metrics\_names)

**Output:**

291/291 - 31s - loss: 0.8313 - accuracy: 0.6416

144/144 - 4s - loss: 0.7781 - accuracy: 0.6566

0.7780812978744507

0.656618595123291

['loss', 'accuracy']

print(X\_train.shape,Y\_train.shape)

print(X\_test.shape,Y\_test.shape)

model = KerasClassifier(build\_fn=createmodel,verbose=0)

epochs = [1, 2]

param\_grid= dict(epochs=epochs)

grid  = GridSearchCV(estimator=model, param\_grid=param\_grid, n\_jobs=1)

grid\_result= grid.fit(X\_train, Y\_train,batch\_size=32)

print("Best: %f using %s" % (grid\_result.best\_score\_, grid\_result.best\_params\_))

(9293, 28) (9293, 3)

(4578, 28) (4578, 3)

Best: 0.674915 using {'epochs': 2}